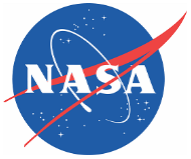


A method to guide assurance for Autonomous Software and Operations

Leila Meshkat

Jet Propulsion Laboratory

California Institute of Technology

Initiative Overview

Description/Goals

- Develop methods and associated tools for making informed decisions for assurance of autonomous software and operations using a modeling approach.
  - Clearly define autonomous software and operations
  - Determine the behaviors that are automated
  - Determine how these behaviors can contribute to system unreliability
  - Determine the questions that need to be addressed in order to manage the system unreliability caused by these behaviors.
  - Develop guidelines for assurance of autonomous software.

Value to NASA

- Currently there are no standard methods for conducting assurance on autonomous software.
- The products of this task will provide the SWA community with clear guidelines on the key risk elements associated with autonomous software and insight regarding areas in the system worthy of further analysis/investigation. It will also provide a means for continuous assurance of the software and commands during the lifecycle of the spacecraft.

FY18 Advancements

- Literature Review
  - Conducted a survey of the literature
- Modeling
  - Generic model of key behaviors and dependencies
  - Customized model to the functions that will be automated for M2020
- Partnerships:
  - Currently working with M2020 project.
    - AI providers, FSW and GDS engineers, SWA manager.
  - Initiated discussions with Fraunhofer

Plans (remainder of FY18)

- Continue development on the three axes:
  - Software Reliability, V&V and Autonomy
    - Literature
    - Assurance Guidelines
  - Modeling
    - Complete qualitative model
    - Start quantifying dependencies for M2020
  - Project Infusion
    - Continue discussions with M2020
    - Iterate on models and explore avenues for direct contribution to FSW reliability and assurance

Description/Goals & Value

Goals

The goal of this proposal is to develop methods and associated tools for making informed decisions for assurance of autonomous software and operations using a modeling approach.

1. Define the key behaviors of autonomous software and operations
2. Develop the root causes for errors for autonomous software and operations.
3. Identify the key dependencies in the system.
4. Create a Bayesian Belief Network (BBN) model of the failure behavior of the system.

a. Develop a methods for eliciting prior probabilities for the BBN using a combination of available data and subject matter expertise.

5. Identify a means for incorporating new observed behavior in the BBN automatically.
6. Develop a use case study that demonstrates the method.
8. Determine assurance guidelines and infusion plan.

Value of the research to NASA

- Currently there are no standard methods for conducting assurance on autonomous software. This is partly due to the fact that flight software is not typically autonomous or responsible for operations and most of our spacecraft are operated using ground based commands.

- The products of this task will provide the SWA community with clear guidelines on the key risk elements associated with autonomous software and insight regarding areas in the system worthy of further analysis/investigation. It will also provide a means for continuous assurance of the software and commands during the lifecycle of the spacecraft.

Literature Review

- Sample Definition:
  - Autonomy software is any software that fits into a part of the spacecraft control process that would normally involve a human. [E. Gat]
    - Attitude control and fault protection software are therefore not autonomy software (at least not on unmanned spacecraft) simply because it is impossible to put a human in those loops.
  - Three necessary conditions for autonomy [Jonsson et. a.]
    1. Autonomy describes a range of behaviors associated with agents, systems that can sense the world around them as well as their own state, can make decisions about what to do, and can carry out their decisions through their own action.
    2. An autonomous system can be controlled by commanding it to achieve a set of goals; the system itself transforms the goals into sequences of actions that accomplish each goal.
    3. An autonomous system flexibly responds to off-nominal situations by adjusting its activity sequence to attain the high-level goals and maintain system safety.
  - In its simplest form, autonomy software consists in control sequences that allow the controlled system to achieve its particular goal while tolerating a certain amount of uncertainty in its environment. [Pecheur]
- Classification Scheme:
  - The levels of autonomy can be classified across the axes of: [R. Knight]
    - Number of decisions that the software makes
    - The complexity of decisions made by the software.
  - The range of behaviors is rule based systems with updates to parameters to self-healing and learning systems.

Literature Review

- Applicable standards:
  - British standards Institute [BSI2017]:
    - Priority areas for standards development for Cars & Autonomous Vehicles identified as:
    - Functional safety, Vehicle communications, cyber resilience, data issues, road and road-side physical infrastructure, vehicle security and road network management.
  - IEEE Standards association has a working group for IEEE P7009, Standard for Fail-Safe Design of Autonomous and Semi-Autonomous Systems.
    - http://sites.ieee.org/sagroups-7009/
    - I have signed up to participate in this working group.
- The degree of autonomy in the detailed design is assessed with regard to the ability of the ground command and control system to intervene in time to support recovery from anomalies in space. [S. Guarro – Aerospace Corporation]

Levels of Autonomy –Cars

- Level 0:
  - The driver controls it all.
- Level 1:
  - Most functions are still controlled by the driver, but a specific function (like steering or accelerating) can be done automatically by the car.
- Level 2:
  - At least one driver assistance system of “both steering and acceleration/deceleration using information about the driving environment is automated.
  - Driver is disengaged from physically operating the vehicle by having his or her hands off the steering wheel AND foot off pedal at the same time.
  - Driver must still always be ready to take control of the vehicle, however.
- Level 3:
  - Drivers are still necessary in level 3 cars, but are able to completely shift "safety-critical functions" to the vehicle, under certain traffic or environmental conditions.
  - The driver is still present and will intervene if necessary, but is not required to monitor the situation in the same way it does for the previous levels.
- Level 4:
  - Level 4 vehicles are "designed to perform all safety-critical driving functions and monitor roadway conditions for an entire trip.
  - However, it's important to note that this is limited to the "operational design domain (ODD)" of the vehicle—meaning it does not cover every driving scenario.
- Level 5:
  - This refers to a fully-autonomous system that expects the vehicle's performance to equal that of a human driver, in every driving scenario—including extreme environments like dirt roads that are unlikely to be navigated by driverless vehicles in the near future.

* [TechRepublic] https://www.techrepublic.com/article/autonomous-driving-levels-0-to-5-understanding-the-differences/

Literature Review

- Methods for V&V of Autonomy Software: [Pecheur]
  - Scenario-Based Testing
    - Software is embedded in test harness that connects to the inputs and outputs of that component, and drives it through a suite of test runs.
    - Each test run is an alternated sequence of provided inputs and expected outputs, corresponding to one scenario of execution of the tested component.
    - An error is signaled when the received output does not meet the expected one.
  - Analytical Verification
    - Analytic verification is the branch of software engineering concerned with establishing, through some mathematically based analysis, that a computer program fulfills a formally expressed requirement. Two main approaches to analytic verification have been developed:
    - Theorem provers build a computer-supported proof of the requirement by logical induction over the structure of the program;
    - Model checkers search all realizable executions of the program for a violation of the requirement.
- Because of the internal complexity of autonomous controllers, and the huge range of situations that they can potentially address, scenario-based testing provides a very limited coverage.
- Model checking can help to find the concurrency problems that would be overlooked in testing, and fix them earlier in the development and thus at cheaper cost.

Literature Review

Ann Marie Neufelder – softrel.com

Bill Taylor, kVA Consulting

Factors that determine reliability of software

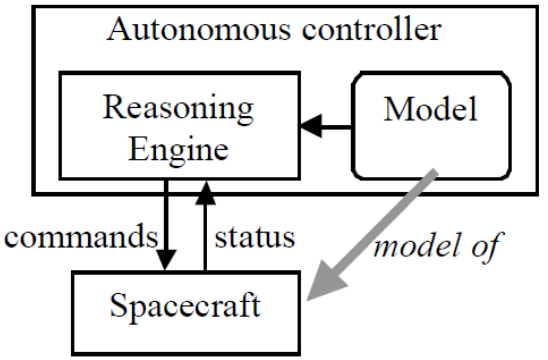
- Safety Assurance for autonomy is based around five key points. We call them the *Five Pillars of Autonomy Assurance*. These are:
  - Safety Culture
  - Autonomous Architecture
  - Safety Process
  - Great Big Data
  - Traceability

Uncertainty Management for Autonomy

- The area that seems more challenging in autonomous software is managing uncertainties associated with:
  - Discontinuities in the state space
    - In non-autonomous control systems, there are typically a sequence of activities or scenarios that are implemented.
    - Autonomous software can jump between sequences in satisfying goals.
  - Situational Awareness
    - There are two types of learning:
      - Collecting new measurements and exercising existing models to determine course of action.
      - Reasoning towards a new model of the world based on observations.
    - The second requires situational awareness and reasoning skills that are dynamic.
- Additional guidelines for assuring autonomy software would need to address specific scenarios where these uncertainties occur.

Literature Review

- Major classes of autonomy software for spacecraft:
  - Planning and Execution
  - Fault Protection and Health Management
  - Distributed Decision Making
  - Science autonomy software
    - Intelligent sensing
    - Opportunistic science
    - Rover navigation software, which can be further broken down into:
      - Terrain analysis and modeling
      - Localization
      - Path generation



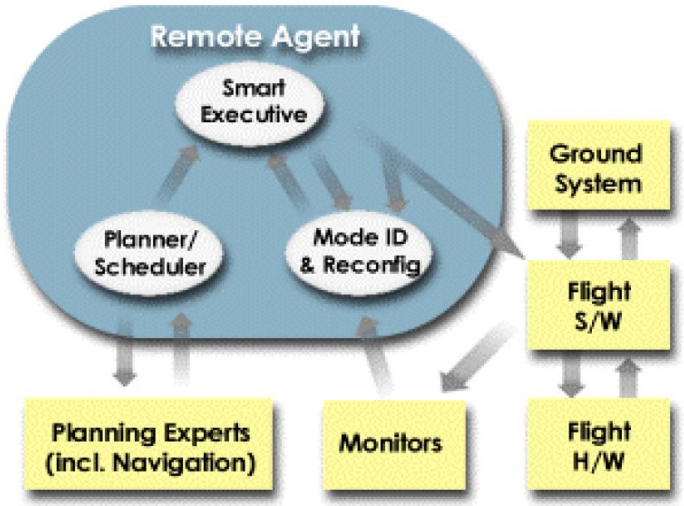
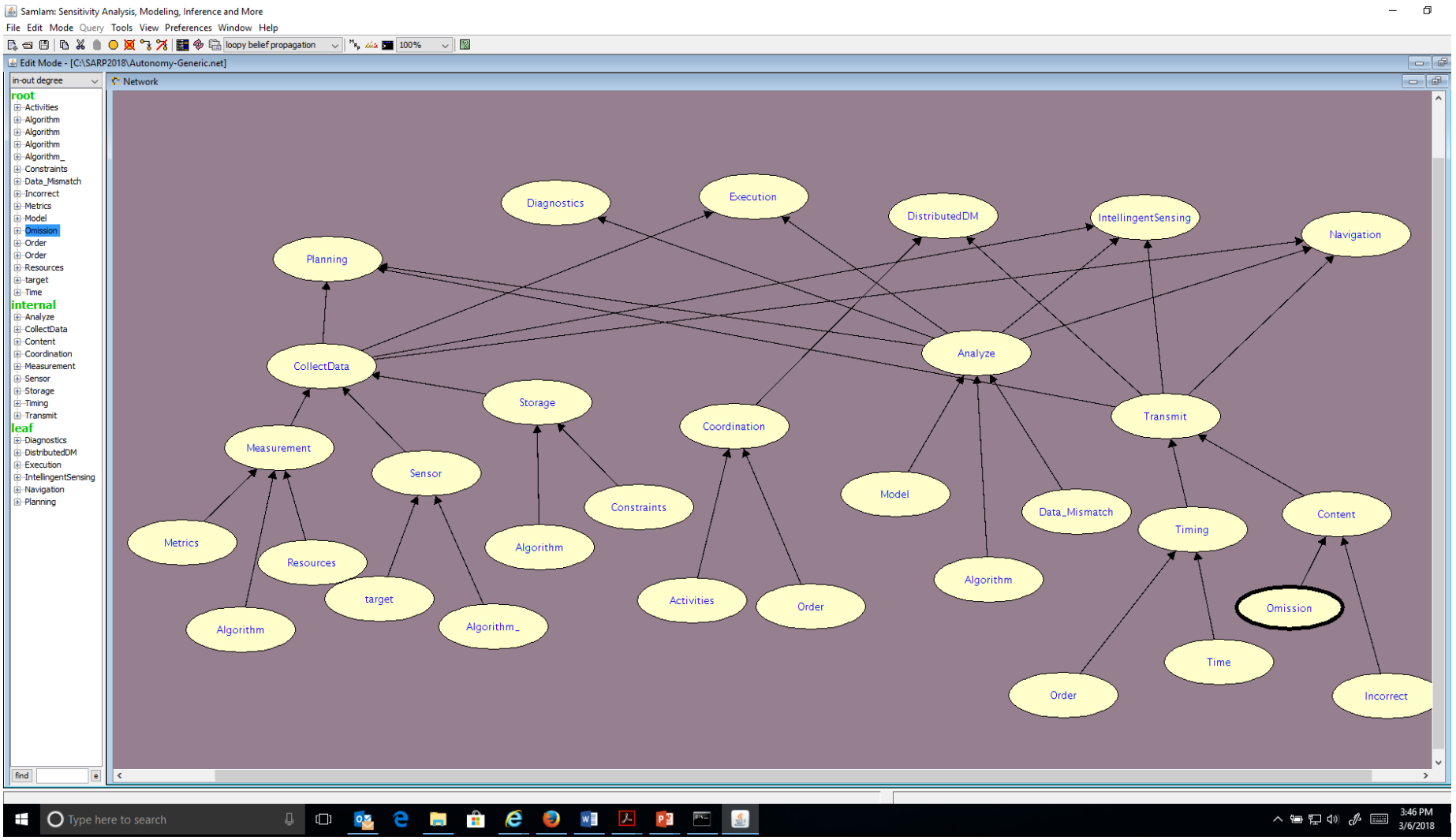
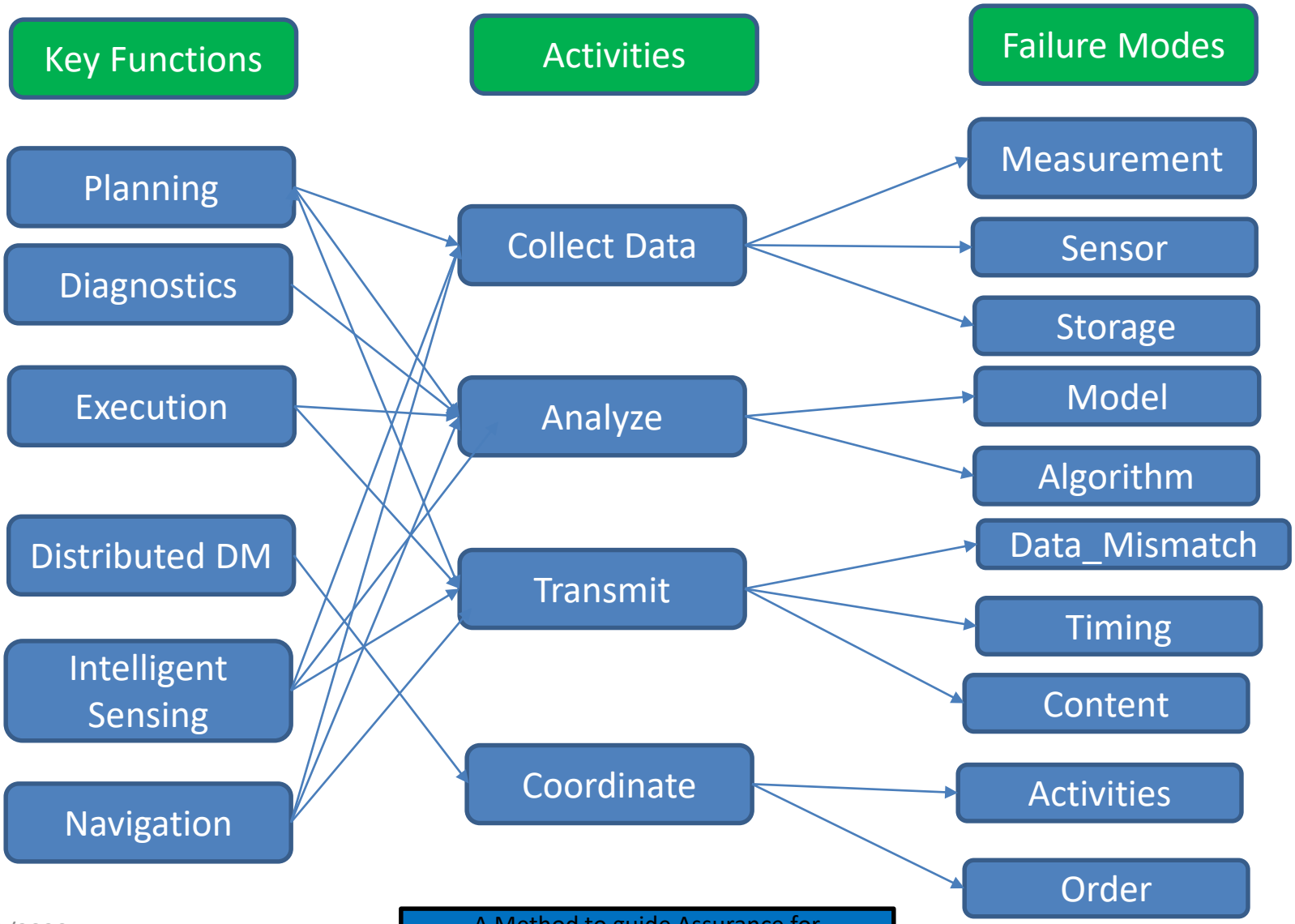


Figure 2. Remote Agent

Dependency Modeling - Generic



Dependency Modeling- Generic



Collaboration / Infusion

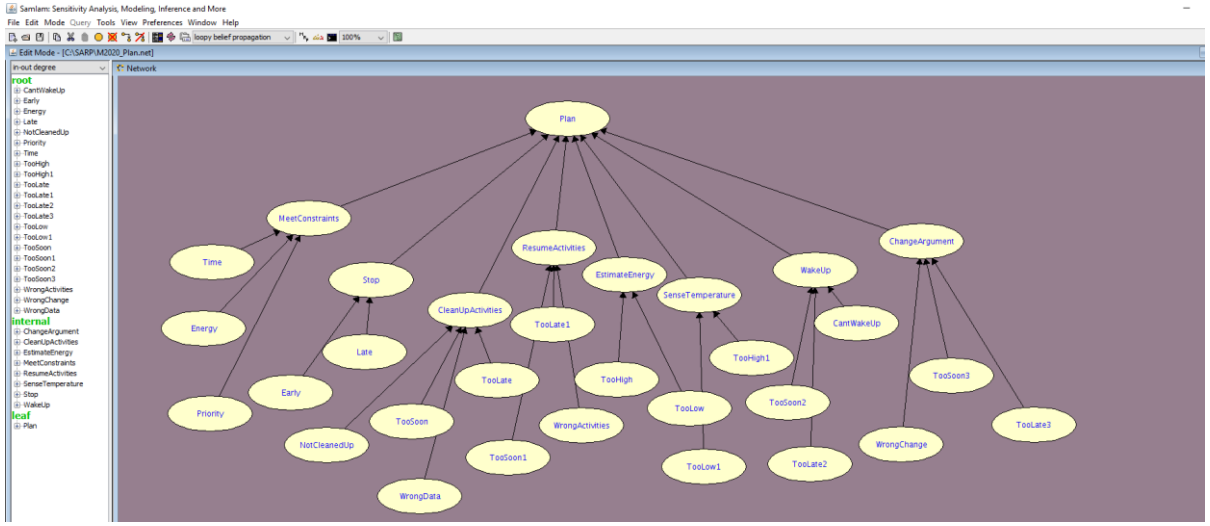
- Looking into the Mars2020 project for infusion opportunities.
  - Collected and synthesized information about autonomy
    - Key requirements
    - Key functions
    - Current plans
  - Developed initial model of:
    - Behaviors associated with these functions
    - Potential failure modes associated with these behaviors.
  - Initiated discussions/collaboration with key project personnel
    - FSW – autonomy – Dan Gaines
    - GDS – James Kurien
    - SWA – Monica Wang
- Looking into synergizing with Autonomy and Mission Operations Assurance activities
  - Preliminary discussion with Autonomy team – Russ Knight
  - Preliminary discussion with Mission Operations Assurance team – Bruce Waggoner
- Initiated discussions with Fraunhofer team for possible collaborations
  - Good synergy between testing, requirements and assurance for autonomy.

Autonomy on M2020

- Key Autonomous Functions
  - Rover Autonomy
    - Autonomous activity execution
    - Opportunistic activity execution
    - Activity pause/resume (around UHF)
    - On-board Battery State-of-Charge estimation
    - Temperature triggered wakeup
  - Remote Science
    - Resolver-based actuator control.
    - Onboard error models for closed-loop control of Remote Sensing Mast (RSM).
    - Vision-based closed-loop targeting correction.
    - Vision-based target selection for Remote Science.

Modeling – M2020

- Using conceptual design of the FSW for both the rover simple planner and remote science to build preliminary dependency models.
  - Conceptual design document has several use case scenarios using which the model is validated.
- Goal is to have a working model which the project agrees on and provide meaningful information to the project for assurance and V&V
- Currently in the process of iterating with the project team.



FY18 Advancements

- Collected and synthesized relevant literature
- Developed modeling framework
- Started working with flight project for use case and infusion
  - Collected and synthesized information about autonomy
    - Key requirements
    - Key functions
    - Current plans
  - Developed initial model of:
    - Behaviors associated with these functions
    - Potential failure modes associated with these behaviors

Plans – FY18 (& beyond)

- Continue research along the three axes:
  - Software Reliability, V&V and Autonomy
  - Dependency Modeling
  - Project Infusion
- Explore ways to validate models.
  - Goal is to produce a model where you can ask the right questions and then use the answers to determine relevant areas of further exploration for the system and metrics for sw assurance.
    - Similar to Ann Maries SW Reliability models.
  - What data should be used for validation?
    - DOD Projects on lab?
- Write a journal quality paper and clear it for external release.
- Explore collaboration opportunities with Fraunhofer

Questions/Discussion

- How can we leverage this research to encourage higher levels and newer kinds of autonomy at NASA?
  - We are very conservative in using autonomy.
  - Testing and assurance seems insufficient, we adopt autonomous behaviors in space very gradually
    - E.g. EO1, Mars rovers
  - Much of the autonomy on M2020 has heritage.
- How can this research be infused in SWA practices?
  - SWA manager on M2020 said she follows the institutional guidelines, and that in order for my research to be infused, it needs to become part of the guidelines.
  - Seems like assurance managers are more in need of new ideas/approaches during critical mission phases such as Operations.

References

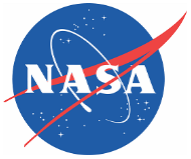
- E. Gat, “Autonomy Software Verification and Validation Might Not Be as Hard as it Seems”, 2004 IEEE Aerospace Conference Proceedings
- A. Jonsson, R. Morris, L. Peterson, “Autonomy in Space Current Capabilities and Future Challenges”, AI Magazine, Volume 28, No. 4.
- J. Kurien and M. R-Moreno, “ Intrinsic Hurdles in Applying Automated Diagnosis and Recovery to Spacecraft”, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans – Special issue on model-based diagnostics, Volume 40, Issue 5, September 2010, Pages 945-958.
- S.B. Guarro, G.A. Johnson-Roth, and W.F. Tosney, June 2010, “Mission Assurance Guide”, The Aerospace Corporation, Technical Operating Report TOR-2007(8546)-6018 Rev. B.
- M. Wetherholt, “ DRAFT Cross Talk Article on NASA Software Assurance”, September 2015, NASA, Washington DC, Document ID: 20150015579

References

- I. Nesnas, CLARAty: Coupled Layer Architecture for Robotic Autonomy, Mars Technology Program Technology Infusion Review
- PAS 754: 2014 " Software trustworthiness Governance and management Specification"
- http://www.softrel.com/ArtificialIntelligenceEnabledSoftwareReliability.html
- http://www.kvausa.com/the-challenge-of-autonomy-safety-assurance/
- DRAFT Cross Talk Article on NASA Software Assurance (https://ntrs.nasa.gov/search.jsp?R=20150015579 2018-02-14T20:48:35+00:00Z
- BSI (British Standard Institut)e and the Transport Systems Catapult, " Connected and autonomous vehicles, A UK standards strategy, Summary report", March 2017
- S. Grenander, K. Simpson and O. Sindiy, " The Autonomy System Architecture", AIAA Space Conference, 2009

References

- C. Pecheur, " Verification and Validation of Autonomy Software at NASA", The NASA STI Program Office NASA/TM-2000-209602
- E. Vassev and M. Hinchey, "Autonomy Requirements Engineering for Space Missions, NASA Monographs in Systems and Software Engineering, Springer International Publishing Switzerland 2014, Chapter 2: Handling Autonomy Requirements for ESA Systems" pgs. 47-103
- P. Schmidt, D. Brueck, M. Rokey, J. Pope, " Independent Assessment of Two NASA Fault Management Software Architectures", The Aerospace Corporation, 2012 NASA Fault Management Workshop, April 10-12, New Orleans, LA.
- [TechRepublic] https://www.techrepublic.com/article/autonomous-driving-levels-0-to-5-understanding-the-differences/
- N.E. Boudette, "Tesla's Self-Driving System Cleared in Deadly Crash," New York Times, 19 Jan. 2017.

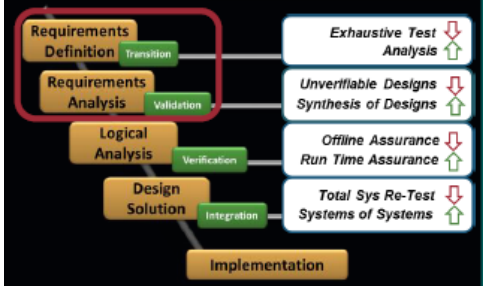


Backup Charts

Verification and Validation of Autonomous Systems
Verifiable Requirements for Complex Systems
Air Force Research Laboratory

Strategy

Providing formal assurance arguments; Increasing trust in the next gen highly complex and autonomous systems



Technical Areas

- Formal Design and Analysis
- Run Time Assurance
- Compositional Systems of Systems Cert
- Synthesis of UAV Mission Plans

Leadership

- Co-Lead OSD Autonomy COI TEVV WG
- Support AFRL Autonomy S&T lead in TEV&V

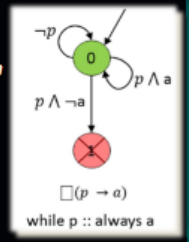
Formalized Requirements

Enables early **Validation** and provides a foundation for architecture and model **Verification**

SpeAR (Specification and Analysis of Requirements)
https://github.com/lgwagner/spear/

- Formal patterns to capture requirements
- Supports 30 KSU Requirements patterns
- AFRL / Rockwell Collins developed additional patterns: *while, initial, delta, range, & transition*

while p :: always a	$\Box(p \rightarrow a)$
while p :: exists a	$\Box(p \rightarrow \Diamond a)$
while p :: a proceeds b	$\Box(p \rightarrow (\neg b\, U ((a \vee \neg p) \vee \Box \neg b)))$
while p :: a responds to b	$\Box(p \rightarrow ((b \rightarrow (p\, U (a \wedge p))) U (\neg p \vee \Box((b \rightarrow (p\, U (a \wedge p))))))$



While the Tank Low Level Sensor is off, the pump shall be on and the valve shall be closed
while (low_sensor == #OFF) :: always (pump_state == #ON and valve_state == #CLOSED)

Accomplishments

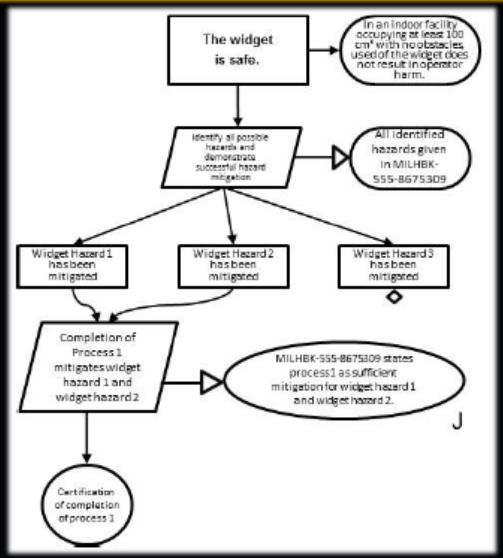
- Refined Patterns: Applied to 90% of AutoGCAS Requirements
- Completed Two Tanks Control Challenge Problem
- SpeAR v2.0 released to public in April 2016

Future Work

- Applying to control oriented UxV autonomy Challenge Problem
- Transition to Lockheed Martin's Quantum Annealing compositional V&V of Control Systems models

Assurance Cases for Certification of Autonomy

Providing evidence of system safety through multiple V&V techniques like M&S, testing, formal verification, and run time assurance



Assurance (Safety) Case is a "…comprehensive and defensible argument that a system is acceptably safe (or secure) to operate in a particular context."
– Professor Tim Kelly, University of York

Approved for Public Release.
Case #: 88ABW-2016-2137

M2020 Mission Success Criteria

- Land safely on the surface of Mars.
- Provide mobility capability on the surface of Mars.
- Operate elements of the science and technology instrument suite integrated on the Mars 2020 Rover mission in a manner that accomplishes the Mars 2020 science and program objectives listed in Section 2.
- Acquire and place on the surface of Mars a suite of scientifically selected and documented samples of Martian materials and standards / blanks adequate to address the mission objectives.
- The returnable samples are at documented locations on Mars such that they would be accessible by a future mission.
- Provide for regular public release of imagery and other science data via the Internet as well as regular releases for public information purposes.
- Archive a copy of verified, validated, and calibrated data acquired by the mission to the Planetary Data System within six months after receipt of the data on Earth.

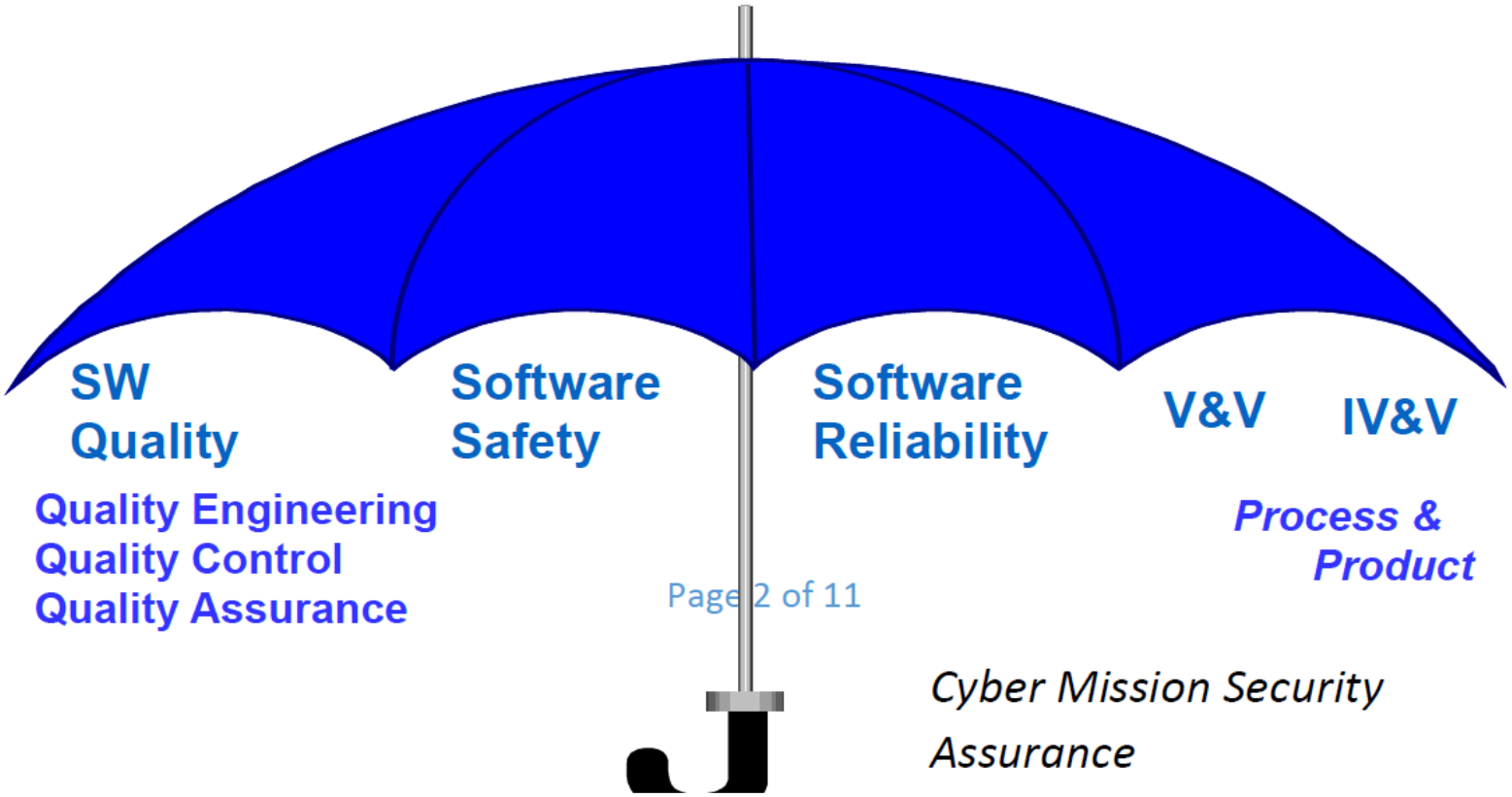


Figure 1. NASA's Software Assurance Umbrella of Risk Mitigation

SW Engineering & Assurance Classes

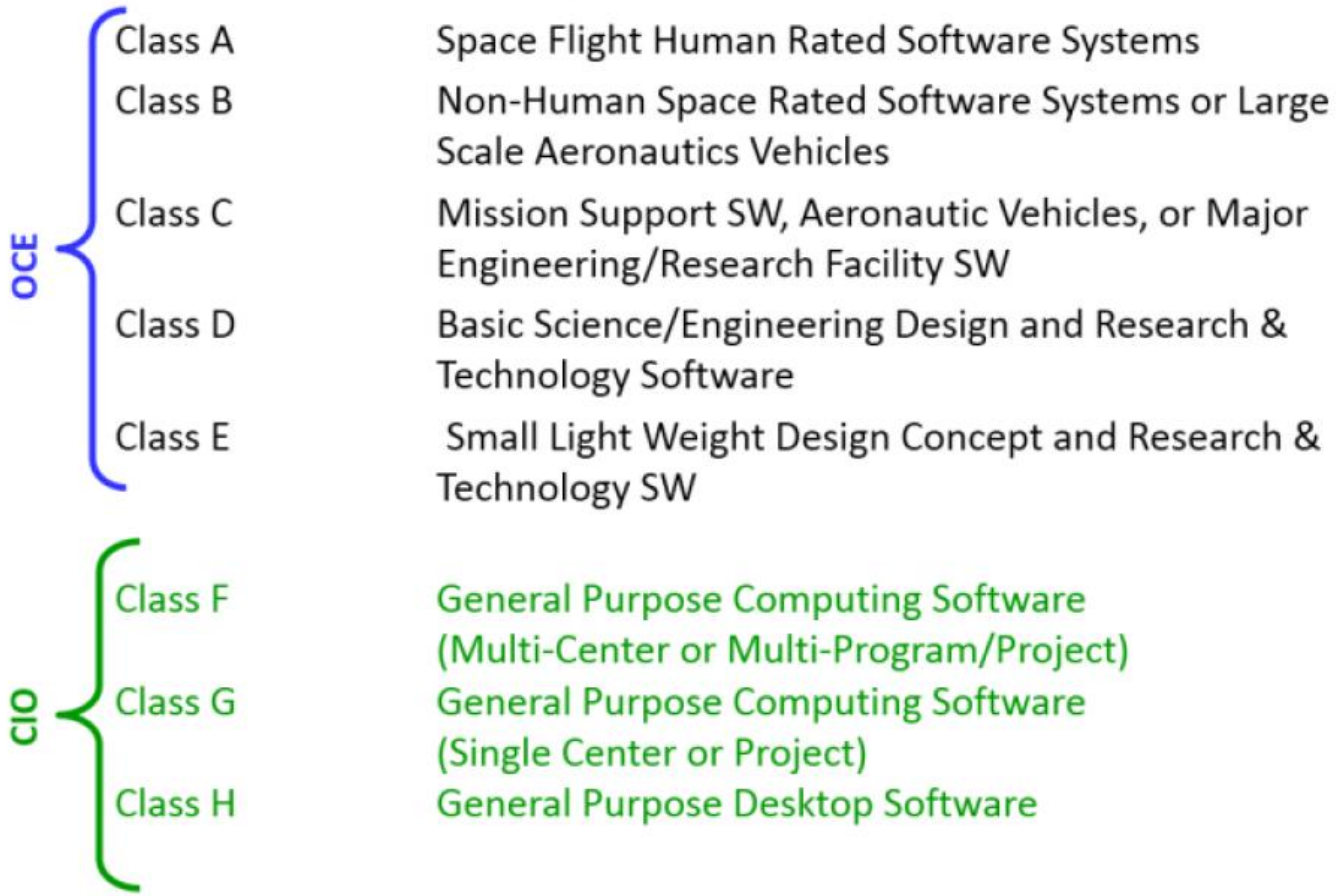


Table 1. NASA Software Classifications